Preconditioner-Based Contact Response and Application to Cataract Surgery

Hadrien Courtecuisse, Jérémie Allard, Christian Duriez, Stéphane Cotin

Preconditioner-Based Contact Response and Application to Cataract Surgery

Hadrien Courtecuisse, Jérémie Allard, Christian Duriez, Stéphane Cotin

Shaman Group, INRIA Lille North Europe

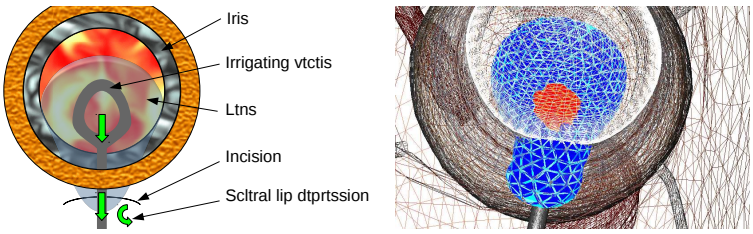
Abstract. In this paper we introduce a new method to compute, in real-time, the physical behavior of several colliding soft-tissues in a surgical simulation. The numerical approach is based on finite element modeling and allows for a fast update of a large number of tetrahedral elements. The speed-up is obtained by the use of a specific preconditioner that is updated at low frequency. The preconditioning enables an optimized computation of both large deformations and precise contact response. Moreover, homogeneous and inhomogeneous tissues are simulated with the same accuracy. Finally, we illustrate our method in a simulation of one step in a cataract surgery procedure, which require to handle contacts with non homogeneous objects precisely.

1 Introduction

Several methods have been presented for real-time bio-mechanical simulations of contacting soft-tissues. However, this work is motivated by a training simulator for cataract surgery, whose needs can not be addressed with existing methods. Thus, we first present the context and the motivations for this work before highlighting the simulation needs and the contributions of our method.

Context: In developed countries, when the crystalline lens is being clouded over by a cataract, a therapy based on *phacoemulsification* is commonly practiced. Training simulation of this procedure has been studied [3,4], and a commercial solution is provided by VRmagic®. Another surgical procedure known as Manual Small Incision Cataract Surgery (MSICS), requires low technology with almost equivalent results when performed by a well-trained specialist. A recent report [7] shows that there is a great need of training for this surgery, to face the huge number of people that need care. This work provides the first steps toward a simulator that could help to solve this training bottleneck.

Unlike during phacoemulsification, MSICS consists in pulling out the crystalline lens as a single piece through a small incision (see Fig. 1). This step generates large deformations on both lens and eyeball. Moreover, for older patients, the center of the lens can be much stiffer than the periphery. This inhomogeneity must be taken into account, as it impacts the successful completion of the surgery. Indeed, it changes the mechanical behaviors and may require adapting the size of the incision. A training system for MSICS necessitates an accurate modeling of these soft-tissues in a simulation executed in real-time.



(a) Extraction of the nucleous (b) Simulated procedure

Fig. 1. Nucleus extraction with an irrigating vectis. (a) motion of the tool, (b) simulated lens. Elements at the center (in red) are stiffer than at the periphery (blue).

Needs and contributions: In this context of tradeoff between accuracy and computation time, we need: (1) to account for precise contact response in highly constrained cases and for strong stiffness inhomogeneities. We note that these inhomogeneities create ill-conditioned systems that are more difficult to solve. (2) to use a large number of nodes and elements in the finite element formulation, in order to maintain the realism of the deformations.

The contribution of this paper is to answer these needs (which probably exist for other kind of surgery) in a real-time simulation. The method is based on an implicit integration scheme (to manage the large range of stiffnesses while keeping large time steps) and uses a preconditioner that is updated at low frequency. This preconditioner provides almost an exact solution to the deformation solver, allowing its fast convergence. Moreover, it provides a very good estimation of the mechanical coupling between the contacting nodes. This allows for a precise constraint solving process based on accurate contact laws.

2 Related Works

Many previous works involve the simulation of soft body in contacts. We will concentrate here on aspects related to the proposed strategies to solve the mechanical and contact equations once they have been expressed.

The core mechanical equations are based on Newton's second law and we rely on an linearized implicit integration scheme [2], which is more stable when sudden contacts occur. Thus we update the next positions and velocities with the following equations :

$$\mathbf{M}\mathbf{a}_{t+h} = \mathbf{f}(\mathbf{x}_{t+h}, \mathbf{v}_{t+h}) \qquad \mathbf{v}_{t+h} = \mathbf{v}_t + h\mathbf{a}_{t+h} \qquad \mathbf{x}_{t+h} = \mathbf{x}_t + h\mathbf{v}_{t+h} \tag{1}$$

$$\underbrace{\left(\mathbf{M} + h\mathbf{B} + h^2\mathbf{K}\right)}_{\mathbf{A}}\mathbf{a}_{t+h} = \underbrace{\mathbf{f}(\mathbf{x}_t, \mathbf{v}_t) + h\mathbf{K}\mathbf{v}_t}_{\mathbf{b}} \tag{2}$$

where $\mathbf{M}$ is the *mass* matrix, $\mathbf{B}$ the *damping* matrix and $\mathbf{K}$ the *stiffness* matrix.

Eq. (2) must be solved at each time-step, but continuously changes due to material and geometrical non-linearities. As the system matrix is positive semi-definite, a popular algorithm to efficiently solve this problem is the Conjugate

Gradient (CG) iterative solver. However, it suffers from convergence issues for ill-conditioned matrices, which can appear for inhomogeneous materials or meshes with varying element sizes. Preconditioning techniques consists in computing an approximation $\mathbf{P}$ of matrix $\mathbf{A}$ which is easier to invert. Then, this preconditioner can be used to solve (2) by relying on $\mathbf{P^{-1}A}$, which is better conditioned and thus converge to an adequate solution in fewer iterations. However, computing a preconditioner adds two overheads: first to invert (or factorize) the preconditioner itself, then to apply it at each CG iteration. Several preconditioners can be used, from simple diagonal matrices [2] to costly but precise incomplete Cholesky factorizations.

When contacts occur, they must be taken into account in the above equation system. A common solution consists in using a penalty method, which handle contacts by adding a contact force $\mathbf{f} = k\delta\mathbf{n}$ at each contact point, where δ is a measure of the interpenetration, $\mathbf{n}$ is the contact normal and k is a stiffness factor. This stiffness is difficult to determine, as it must be tuned according to the stiffness of the object in contact. This becomes particularly an issue when dealing with inhomogeneous objects. Stiff penalty forces can either be considered as explicitly-integrated external forces, which can introduce instabilities and would require lowering the time-steps. Alternatively, they can be placed in the same implicit equations as the internal forces, creating a very large and evolving system that can be difficult to solve.

Other methods rely on Lagrange multiplier [8,10,6] to compute contact forces such that all intersections are removed at the end of each time step. The core computation of the algorithm involves solving a Linear Complementary Problem (LCP) deriving from Signorini's law :

$$\boldsymbol{\delta} = \mathbf{HCH}^T\boldsymbol{\lambda} + \boldsymbol{\delta}_0 \qquad \text{with} \qquad 0 \leq \boldsymbol{\delta} \perp \boldsymbol{\lambda} \geq 0 \tag{3}$$

where $\mathbf{H}$ is the Jacobian of the contacts, and $\mathbf{C}$ is the *compliance* matrix, λ is the contact force, $\boldsymbol{\delta}_0$ and $\boldsymbol{\delta}$ are the measure of interpenetrations before and after collision response. This equation means that, if λ is positive at the end of the time step, then δ must be equal to zero, and vice versa.

For explicit methods $\mathbf{C}$ is the inverse of the (often diagonalized) mass matrix, whereas for implicit schemes it also involves damping and stiffness, i.e. $\mathbf{C} = (\frac{1}{h^2}\mathbf{M} + \frac{1}{h}\mathbf{B} + \mathbf{K})^{-1}$ for the scheme used in (2). This matrix changes at every time steps, and its computation can be prohibitive for large deformable meshes. It is possible to use an approximation of the compliance, because an approximated local deformation in response to a collision is visually acceptable. The most extreme approximation is to only consider the mass, as for explicit schemes. When it is used, contacts are corrected without any mechanical coupling between nodes. A more accurate method has been proposed in [11], inspired by the co-rotational formulation used in FEM to remove non-linearities introduced by rotations. $\mathbf{C}$ can be precomputed from the rest configuration and updated at each time step based on a local estimation of the rotations. However, this approximation can become inaccurate for large deformations. Moreover, it requires storing a dense matrix, with $9n^2$ values where n is the number of vertices of the object, therefore preventing its application on detailed meshes.

3 Inhomogeneous Deformable Model on GPU

Soft tissue deformations are modeled using a geometrically non-linear elastic formulation and a Finite Element Method (co-rotational model introduced in [9]). The local rotation of each element is estimated in order remove the influence of geometrical nonlinearities, allowing to account for large deformations. To compute the deformation in real-time with detailed meshes (more than 10,000 tetrahedral elements), we implemented a CG solver on the GPU [1].

Recently, we introduced a new approach [5] to maintain a good approximation of the inverse of the system matrix during the simulation. This method exploits time coherency to update an exact factorization of the system matrix within a separated loop computed at a lower frequency. It only requires a few simulation steps to provide a new factorization, thus constantly providing a good approximation usable as a preconditioner. It is further improved by estimating rotations between the current position and the state used for the last factorization, similar to the co-rotational formulation. Even if the preconditioner is applied on the CPU, the GPU can still be used for the remaining operations. Finally, as it uses a sparse factorization, it supports simulations with a large number of elements.

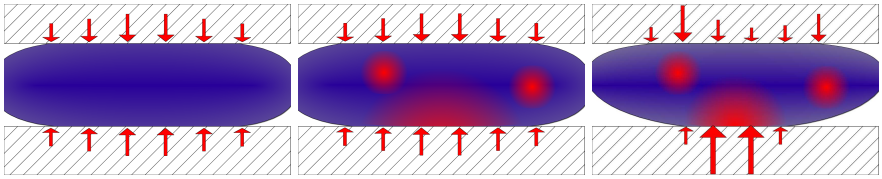
As a consequence the above method significantly reduces the number of iterations needed for the CG algorithm to converge. However, the paper only considered simulations involving a single object without any contacts. In this paper, we extend this idea to the computation of contact responses, as is detailed in the next section, and apply this new technique in the field of medical simulation. In the context of the cataract simulation, two deformable tissues need to be simulated: the eye lens and sclera (Fig. 1) which both undergo large deformations combined with multiple contacts during surgery. Furthermore, the crystalline lens is by nature inhomogeneous, with a stiff kernel and softer boundaries, while the meshes of these structures contain elements of potentially very different size (in particular near the incision located in the sclera). All of these characteristics lead to a very poor conditioning of the system matrix, and it is obvious that in this case, preconditioning techniques could greatly improve the converge rate of the solver. For this application, we found the $\mathbf{LDL}^{\mathrm{T}}$ factorization to be more stable than the $\mathbf{LL}^{\mathrm{T}}$ Cholesky factorization. Therefore, the complete expression of the preconditioner we use is:

$$\mathbf{P} = \mathbf{R}_{t\rightarrow t-\Delta t}^{\mathrm{T}} \ \mathbf{L_{t-\Delta t}} \mathbf{D_{t-\Delta t}} \mathbf{L_{t-\Delta t}^{T}} \ \mathbf{R}_{t\rightarrow t-\Delta t} \tag{4}$$

where $\mathbf{R_{t\rightarrow t-\Delta t}}$ is the current local rotations matrix, and $\mathbf{L_{t-\Delta t}} \mathbf{D_{t-\Delta t}} \mathbf{L_{t-\Delta t}^{T}}$ the most recent factorization.

4 Preconditioner for Contact Response

Local mechanical coupling must be taken into account to compute accurate collision response (see Fig 2). Indeed, while the constraint-based formulation will ensure a contact-free configuration at the end of each time-step in all cases, the local



(a) homogeneous object (b) inhomogeneous object without coupling (c) inhomogeneous object with coupling

Fig. 2. Force repartition for homogeneous and inhomogeneous objects on which collisions are solved with and without mechanical coupling. Stiffer part are shown in red.

deformation near contacts obtained without the correct compliance (Fig. 2(b)) is invalid. While this difference would be reduced after several time-steps, using a more accurate compliance produces better intermediate configurations and, more importantly for real-time simulations, allow larger time-steps.

To solve this issue, we propose to extend the work introduced in [11], using the preconditioner as compliance matrix instead of a precomputed dense inverse matrix. Indeed, the compliance matrix is only a scaled version of the inverse system matrix: $\mathbf{C} = h^2\mathbf{A}^{-1}$. Moreover, we showed in section 3 that we maintain a good approximation of $\mathbf{A}^{-1}$ by updating the preconditioner at low frequency. Thus, if $\mathbf{P}$ remains a good approximation of the system, it can be re-used to build the compliance matrix. That way, we guarantee a contact-free configuration at the end of the time step with almost the exact mechanical coupling of elements taken into account. To build the LCP matrix, we combine eq. (3) and (4) to compute for each object:

$$\begin{aligned} \mathbf{HCH}^{\mathrm{T}} = h^2\,\mathbf{HA}^{-1}\mathbf{H}^{\mathrm{T}} &\approx\; h^2\,\mathbf{H}\left(\mathbf{RLDL}^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}\right)^{-1}\mathbf{H}^{\mathrm{T}} \\ &\approx h^2\,\mathbf{J}\left(\mathbf{LDL}^{\mathrm{T}}\right)^{-1}\mathbf{J}^{\mathrm{T}} \qquad \text{with} \quad \mathbf{J} = \mathbf{HR} \end{aligned} \tag{5}$$

This computation can be implemented in three steps. The first step consists in applying the local rotations since the last update to $\mathbf{H}$. This operation is inexpensive because it is done by computing the product of a block-diagonal matrix $\mathbf{R}$ with a sparse matrix into matrix $\mathbf{J}$ with the same sparsity structure. Next, we compute the product of $\mathbf{J}^{\mathrm{T}}$ with the inverse of the factorization. This is achieved by computing rows independently, each requiring two sparse triangular solves (STS) using one row of $\mathbf{J}$. Finally, the resulting matrix is multiplied by matrix $\mathbf{J}$ to obtain the final contribution to the LCP matrix.

The number of STS in step 2 is proportional to the number of contacts. They can therefore become expensive in complex scenarios. However, each row is independent. Thus, we propose to implement them on GPU, by computing several STS in parallel, reducing the level of parallelism that must be achieved within each STS. This maps nicely to the two-level SIMD architecture of todays GPU where synchronizations within a group of cores is fast, whereas global synchronization over multiple groups is much more prohibitive. Such synchronizations are necessary to guarantee the respect of dependencies within the computation,

and would be difficult to implement efficiently if only a single STS was computed on all GPU computing cores.

Matrices **H**, **J** and **L** can be stored in *compressed sparse row* (CSR) format to save memory. However in the GPU implementation it is more efficient to use a dense representation for **J**, such that the elements of the right-hand vectors for each solve can be accessed directly. Another consequence is that the final matrix product then involves two dense matrices, and can therefore be implemented using a standard BLAS library, such as CUBLAS for NVIDIA® GPUs. While the LCP matrix is computed on GPU, it is then solved on CPU using the traditional Gauss-Seidel algorithm. The size of this matrix only depends on the number of contacts, which is often much smaller that the number of mechanical vertices, so this transfer and solve is much faster that the previous steps in most simulations, and the gain to implement them on GPU would be negligible.

5 Results

All the following results were obtained on a quad-core Intel® Core™ i7 3.07 GHz CPU with a Nvidia® GeForce® GTX 580 GPU.

Evaluation: We measured the error introduced by using an approximation to the compliance, compared to an exact factorization at each time step. We produced a simulation involving a non homogeneous lens which is pushed by a sphere through a small hole. The difference of vertices position was compared using Root Mean Square error (see Fig. 3), when using only the diagonal matrix and our method with several update frequencies, with *no update* being equivalent to [11], and *async* corresponding to our contribution where the preconditioner is updated as soon as possible using another thread. Results show that using inaccurate approximation does not lead to the correct behavior. Using a diagonal compliance does not consider the coupling between vertices within each time-step and is stable only for very small steps, relying on the mechanical computations

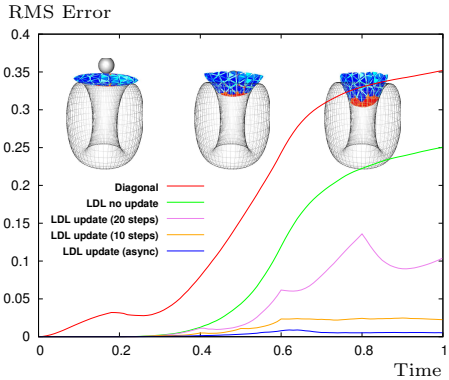


Fig. 3. RMS error of positions over time, introduced by several approximation as compared to an exact factorization.

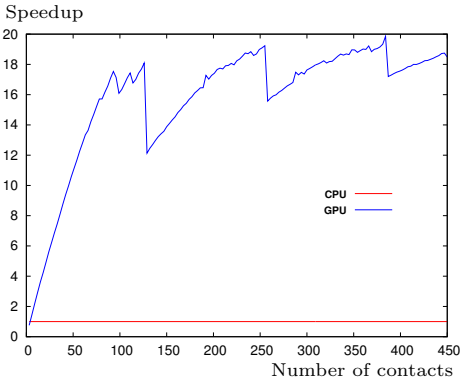


Fig. 4. Speedup of GPU-based algorithm to build the compliance matrix according to the number of contacts.

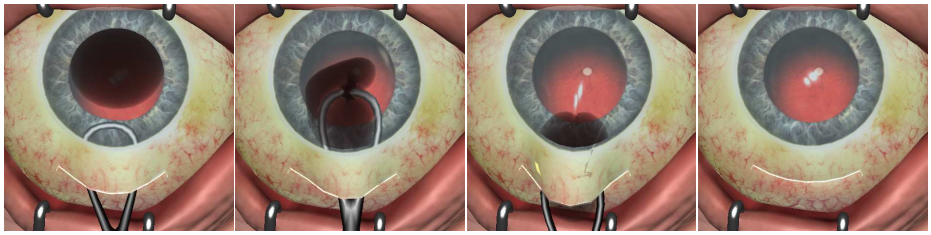


Fig. 5. Simulation of the extraction of the eye lens during MSICS

to propagate contact forces. With the proposed method, we can simulate such scenario in real time if the preconditioner can be updated sufficiently frequently. Indeed, when the factorization is updated more often, the obtained shape and behavior is increasingly accurate. In this example, the multi threaded version updates the preconditioner on average every 3 steps, allowing to simulate almost the exact behavior.

GPU implementation: We measured the computation time (see Fig. 4) required to build the compliance matrix with our GPU-based algorithm compared to a sequential CPU implementation. As expected, the computation time to obtain a single solve with our GPU algorithm (21.90 ms) is larger than the CPU (2.44 ms) version. Thus, we maintain the application of the preconditioner during the mechanical CG solver on the CPU (as each iteration only requires one solve). However, the computation time for solving multiple contacts remains almost constant with the GPU algorithm, which allows to quickly take advantage of the graphic processor. Indeed, until 130 contacts the computation units of the GPU are not fully exploited, and each STS is computed in parallel. Beyond this number of contacts, some of the computation units will compute several STS successively. However, the GPU is able to overlay waiting times, due to synchronizations and read/write operations in memory, with computations for another STS. Thus, solving 260 contacts is only 1.5 times slower than 130.

MSICS Simulation: Finally, we show in Fig. 5 our simulation of the extraction of a diseased lens during a MSICS simulation. The lens is modeled with 1113 vertices and 4862 tetrahedra, whereas the eye contains 1249 vertices and 3734 tetrahedra. The center of the lens is five times stiffer than the periphery. For both organs, we used the LDL-based preconditioner, which was updated on average every 4 time steps for the eye and 3 time steps for the lens. With this preconditioner, the CG required an average of 8.5 iterations to converge.

The lens is removed with the help of the deformations of the organs and friction between the tool and the lens. The surgical instrument is controlled by the user with an haptic device, with the method introduced in [12]. As we model the compliance of the objects in contact with the instrument, we are able to compute and transmit feedback forces to the device.

The simulation runs between 15 and 50 FPS depending on the number of contacts, which varies between 50 and 90 during the extraction. The computation within a single time is split between 31.5% for the preconditioned CG, 47.40%

for the GPU-based assembly of the compliance matrix, and less than 22% for the other computations, including collision detection and LCP solver.

6 Conclusion

In this paper, we have presented a novel method for solving accurate contacts response between soft tissues which is based on the use of a periodically updated preconditioner. The proposed method allows to simulate a large number of elements and supports both homogeneous and inhomogeneous objects with a similar accuracy, while taking into account the mechanical coupling between contacts. We applied our method to a MSICS simulation, which presents very constrained situations during the eye lens extraction. Although this method was illustrated in a cataract surgery, it can be applicable to any simulations with similar requirements. As future works, we would like to simulate the complete simulation of MSICS, including the incision and capsulorhexis, as well as handling the influence of fluid pressure within the eye.

References

1. Allard, J., Courtecuisse, H., Faure, F.: Implicit FEM solver on GPU for interactive deformation simulation. In: GPU Computing Gems Jade Edition, chap. 21. Elsevier (2011)
2. Baraff, D., Witkin, A.: Large steps in cloth simulation. In: Proc. of SIGGRAPH 98. pp. 43–54. ACM (1998)
3. Choi, K.S., Soo, S., Chung, F.L.: A virtual training simulator for learning cataract surgery with phacoemulsification. Comput. Biol. Med. 39(11), 1020–1031 (2009)
4. Comas, O., Cotin, S., Duriez, C.: A shell model for real-time simulation of intraocular implant deployment. In: Bello, F., Cotin, S. (eds.) Biomedical Simulation, LNCS, vol. 5958, pp. 160–170. Springer (2010)
5. Courtecuisse, H., Allard, J., Duriez, C., Cotin, S.: Asynchronous preconditioners for efficient solving of non-linear deformations. In: Erleben, K., Bender, J., Teschner, M. (eds.) VRIPHYS. pp. 59–68. Eurographics Association (2010)
6. Galoppo, N., Otaduy, M., Mecklenburg, P., Gross, M., Lin, M.: Fast simulation of deformable models in contact using dynamic deformation textures. In: Cani, M.P., O'Brien, J. (eds.) SCA. pp. 73–82. Eurographics Association (2006)
7. HelpMeSee, http://helpmesee.org/about/statement-of-purpose/
8. Lenoir, J., Grisoni, L., Meseure, P., Rémion, Y., Chaillou, C.: Smooth constraints for spline variational modeling. In: GRAPHITE '04. pp. 58–64. ACM (2004)
9. Nesme, M., Payan, Y., Faure, F.: Efficient, physically plausible finite elements. pp. 77–80. Eurographics Association (2005)
10. Popescu, D., Compton, M.: A model for efficient and accurate interaction with elastic objects in haptic virtual environments. In: GRAPHITE '03. pp. 245–250. ACM (2003)
11. Saupin, G., Duriez, C., Cotin, S., Grisoni, L.: Efficient contact modeling using compliance warping. In: Computer Graphics International Conference (2008)
12. Saupin, G., Duriez, C., Cotin, S.: Contact model for haptic medical simulations. In: Bello, F., Edwards, P. (eds.) Biomedical Simulation, LNCS, vol. 5104, pp. 157–165. Springer (2008)